Paper No. 16

UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

_____

Ex parte NAWAF K. BITAR, ROBERT M. ENGLISH
and RAJAGOPAL ANANTHANARAYANAN

_____

Appeal No. 2002-0792
Application 08/801,646

_____

ON BRIEF

_____

Before KRASS, FLEMING, and DIXON, **Administrative Patent Judges.**

FLEMING, **Administrative Patent Judge.**

DECISION ON APPEAL

This is a decision on appeal from the final rejection of claims 1 through 6, 13, 14, 18 through 34, 40 through 46, and 52 through 55. Claims 7 through 12, 15 through 17, 35 through 39, and 47 through 51 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in

independent form including all of the limitations of the base claim and any intervening claims.

Invention

The invention relates to a method of scheduling parallel processes in a distributed, multi-kernel, multiprocessor system. In particular, the present invention relates to a system and method for scheduling parallel processes with no kernel-to-kernel communication. A thread model of program execution has proven to be a viable method for parallel execution of program code both in single and multiprocessor machines. Under the thread model, programs are partitioned into a set of parallel activities. Each activity during execution of the program code is called a thread. See page 1 of Appellants' specification.

The problem with the prior art systems is that if scheduling of such a multi-threaded program requires excessive kernel-to-kernel communication, the performance of the program will suffer. Appellants' invention solves this problem by using nanothreads. A shared arena is provided in the user memory, wherein the shared arena includes a register save area for each of the plurality of threads. See page 9 of Appellants' specification.

Appellants' claim 1 is representative of the claimed invention and is reproduced as follows:

1. In a computing system having a processor, a memory and a kernel level scheduler, wherein the processor includes a user mode and a protected kernel mode, a method of scheduling a plurality of threads from a multi-threaded program for execution in user mode, wherein the multi-threaded program includes a user level scheduler, the method comprising the steps of:

defining a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads; and

selecting, at the user level scheduler, a thread from the plurality of threads to be executed on the processor, wherein the step of selecting includes the step of reading register context associated with the selected thread from one of the plurality of register save areas.

References

The references relied on by the Examiner are as follows:

Anderson et al. (Anderson), "Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism", Department of Computer Science and Engineering, University of Washington, Seattle, WA, pp 95-109 (1991).

Polychronopoulos et al. (Polychronopoulos), "Nano-Threads: A User-Level Threads Architecture", CSRD TR 1297, 1993, pp 2-22.

Rejections at Issue

Claims 1 through 6, 13, 14, 18 through 34, 40 through 46, and 52 through 55 stand rejected under 35 U.S.C. § 103 as being unpatentable over Anderson in view of Polychronopoulos.

OPINION

With full consideration being given to the subject matter on appeal, the Examiner's rejections and the arguments of Appellants and the Examiner, for the reasons stated **infra**, we reverse the Examiner's rejection of claims 1 through 6, 13, 14, 18 through 34, 40 through 46, and 52 through 55 under 35 U.S.C. § 103.

In rejecting claims under 35 U.S.C. § 103, the Examiner bears the initial burden of establishing a **prima facie** case of obviousness. **In re Oetiker**, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). **See also In re Piasecki**, 745 F.2d 1468, 1472, 223 USPQ 785, 788 (Fed. Cir. 1984). The Examiner can satisfy this burden by showing that some objective teaching in the prior art or knowledge generally available to one of ordinary skill in the art suggests the claimed subject matter. **In re Fine**, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). Only if this initial burden is met does the burden of coming forward with evidence or argument shift to the Appellants. **Oetiker**, 977 F.2d at 1445, 24 USPQ2d at 1444. **See also Piasecki**, 745 F.2d at 1472, 223 USPQ at 788.

An obviousness analysis commences with a review and consideration of all the pertinent evidence and arguments. "In reviewing the [E]xaminer's decision on appeal, the Board must necessarily weigh all of the evidence and argument." **Oetiker**, 977 F.2d at 1445, 24 USPQ2d at 1444. "[T]he Board must not only assure that the requisite findings are made, based on evidence of record, but must also explain the reasoning by which the findings are deemed to support the agency's conclusion." **In re Lee**, 277 F.3d 1338, 1344, 61 USPQ2d 1430, 1434 (Fed. Cir. 2002). With these principles in mind, we commence review of the pertinent evidence and arguments of Appellants and Examiner.

Appellants argue that for claims 1, 19 and 23, the Examiner has failed to show that Polychronopoulos teaches a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads. Appellants also argue that neither Anderson nor Polychronopoulos teaches a shared arena including a register save area for a plurality of threads or reading register context associated with a selected thread of plurality of register save areas as recited in Appellants' claims 1, 19 and 23. See page 8 of Appellants'

brief. We note that Appellants' claim 1 recites:

> defining a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads; and
>
> selecting, at the user level scheduler, a thread from the plurality of threads to be executed on the processor, wherein the step of selecting includes the step of reading context associated with the selected thread from one of the plurality of register save areas.

We also note that Appellants' claim 19 recites:

> first program code executing in the processor for creating a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads; and
>
> second program code executing in the processor, wherein the second program code includes scheduling code for scheduling threads from the plurality of threads, wherein the scheduling code includes program code for selecting a thread from the plurality of threads and for switching to the selected thread by reading register context associated with the selected thread from one of the plurality of register save areas.

Appellants' claim 23 recites:

> first program code executing in one of the plurality of processors for creating a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads; and
>
> second program code executing in the processor, wherein the second program code includes scheduling code for scheduling threads from the plurality of threads, wherein the scheduling code includes program code for selecting a thread from the plurality of threads and for switching to the selected thread by reading register context associated with the selected thread from one of the plurality of register save areas.

The Examiner states that Anderson does not expressly teach a shared arena within the memory, wherein the shared arena includes a register save area and selecting includes the step of reading register context associated with the selected thread from one of the plurality of register save areas. See page 3 of the answer. The Examiner relies on Polychronopoulos for this teaching. The Examiner points us to page 7 and page 9 of Polychronopoulos.

Upon our review of Polychronopoulos, we find that in section 3 titled "Scheduler-Kernel Interfaces" found on page 6, Polychronopoulos teaches that in order to properly support the user-level scheduling model, a set of communication points between the user-level scheduler and the kernel must be defined. Polychronopoulos then further states that these interfaces will allow the user-level scheduler to request/release processor resources and reorder run queues. Polychronopoulos then further states that the reference only defines the interface semantic and does not specify the implementation of the communication mechanism. Polychronopoulos then suggests that some implementations may choose to use a block of shared memory used between the user-level scheduler and the kernel as the communication venue while other implementations may be an explicit upcall mechanism as described in Anderson. We agree

with the Examiner that the suggestion to use shared memory between the user-level scheduler and the kernel as a communication venue is of interest. However, we fail to find that Polychronopoulos has described any further details of such a communication mechanism other than this one suggestion. Therefore, we fail to find that Polychronopoulos teaches the above limitations as recited in Appellants' claims 1, 19 and 23.

Appellants also argue that Polychronopoulos and Anderson fail to teach the use of shared arena as a communication mechanism for conveying register context in the scheduling of threads and the use of a number requested variable and a number allocated variable, both of which are stored in the shared arena, to allocate threads to the processor as recited in Appellants' claim 13. See page 10 of the brief.

We note that Appellants' claim 13 recite:

> defining a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads;
>
> starting the program, wherein the step of starting the program includes the step of setting, via the user level scheduler, a number requested variable within the shared arena requesting that one or more processors from the plurality of processors be assigned to the program and setting, via the kernel level scheduler, a number allocated variable within the shared arena indicating how many processors from the plurality of processors are assigned to the program; and

> allocating, at the user level scheduler, one or more threads of the plurality of threads to each of the processors assigned to the program, wherein the step of allocating includes the step of reading register context from one of the plurality of register save areas.

For the reasons as we have pointed out above, we fail to find that Polychronopoulos or Anderson teaches the above limitations.

For claims 27, 40 and 52, Appellants argue that the Examiner has failed to show how the references cite and teach or suggest all the limitations recited in these claims. In particular, Appellants argue that the Examiner has not shown how the references teach the use of a first and second kernel of scheduler to schedule threads across a first and second kernel.

We note that Appellants' claim 23 recites:

> a kernel level scheduler, executing within the protected kernel mode of one of the plurality of processors, for allocating processors to a program;
>
> first program code executing in one of the plurality of processors for creating a shared arena within the memory, wherein the shared arena includes a register save area for each of the plurality of threads; and
>
> second program code executing in the processor, wherein the second program code includes scheduling code for scheduling threads from the plurality of threads, wherein the scheduling code includes program code for selecting a thread from the plurality of threads and for switching to the selected thread by reading register context associated with the selected thread from one of the plurality of register save areas.

We note that Appellants claims 40 recites:

> allocating a shared arena within the memory, wherein the shared arena is accessible to said first and second kernel level schedulers and wherein the shared arena includes a user-level run queue;
>
> setting a number requested variable within the shared arena requesting that one or more processors from the plurality of processors be assigned to process the plurality of threads;
>
> setting, via the first kernel level scheduler, a number allocated variable within the shared arena indicating the number of first processors that are assigned to process the plurality of threads;
>
> selecting one or more of the plurality of threads from the user-level run queue;
>
> assigning one of the plurality of threads selected from the user-level run queue to each of the assigned first processors;
>
> adding to the number allocated variable within the shared arena a number indicating the number of second processors that are assigned to process the plurality of threads;
>
> selecting one or more of the plurality of threads from the user-level run queue.

Appellants' claim 52 recites:

> allocating a first and a second shared arena within the memory, wherein each shared arena includes a register save area;
>
> forming a queue of threads from the first program;
>
> forming a queue of threads from the second program;
>
> setting a number requested variable within the first

shared arena requesting that one or more processors from the plurality of processors be assigned to process the plurality of threads from the first program;

setting, via the first kernel level scheduler, a number allocated variable within the first shared arena indicating the number of first processors that are assigned to process the plurality of threads from the first program;

assigning one or more threads of the plurality of threads from the first program to each of the assigned first processors, wherein the step of assigning includes the step of reading register context from one of the plurality of register save areas within the first shared arena;

adding to the number allocated variable within the first shared arena a number indicating the number of second processors that are assigned to process the plurality of threads from the first program;

assigning one or more threads of the plurality of threads from the first program to each of the assigned second processors, wherein the step of assigning includes the step of reading register context from one of the plurality of register save areas within the first shared arena;

setting a number requested variable within the second shared arena requesting that one or more processors from the plurality of processors be assigned to process the plurality of threads from the second program;

setting, via the first kernel level scheduler, a number allocated variable within the second shared arena indicating the number of first processors that are assigned to process the plurality of threads from the second program; and

assigning one or more threads of the plurality of threads from the second program to each of the assigned first processors, wherein the step of assigning includes the step of reading register context from one of the plurality of register save areas within the second shared arena.

We fail to find that the Examiner has shown that Anderson or Polychronopoulos teaches these above limitations.

In view of the foregoing, we have not sustained the Examiner's rejection of claims 1 through 6, 13, 14, 18 through 34, 40 through 46, and 52 through 55 under 35 U.S.C. § 103 as being unpatentable over Anderson in view of Polychronopoulos.

REVERSED

ERROL A. KRASS Administrative Patent Judge	)	
MICHAEL R. FLEMING Administrative Patent Judge	)	BOARD OF PATENT APPEALS AND INTERFERENCES
JOSEPH L. DIXON Administrative Patent Judge	)	

MRF:pgc

Appeal No. 2002-0792
Application 08/801,646

Schwegman Lundberg Woessner & Kluth
P.O. Box 2938
Minneapolis, MN 55402